SHREC'20 track: Retrieval of digital surfaces with similar geometric reliefs

Elia Moscoso Thompson[a], Silvia Biasotti[a], Andrea Giachetti[b], Claudio Tortorici[c], Naoufel Werghi[d], Ahmad Shaker Obeid[d], Stefano Berretti[e], Hoang-Phuc Nguyen-Dinh[f,g], Minh-Quan Le[f,g], Hai-Dang Nguyen[f,g], Minh-Triet Tran[f,g], Leonardo Gigli[h], Santiago Velasco-Forero[h], Beatriz Marcotegui[h], Ivan Sipiran[i], Benjamin Bustos[j], Ioannis Romanelis[k], Vlassis Fotis[k], Gerasimos Arvanitis[k], Konstantinos Moustakas[k], Ekpo Otu[l], Reyer Zwiggelaar[l], David Hunter[l], Yonghuai Liu[m], Yoko Arteaga[n,o], and Ramamoorthy Luxman[p]

[a]Istituto di Matematica Applicata e Tecnologie Informatiche 'E. Magenes' - CNR
[b]Department of Computer Science, University of Verona
[c]Technology Innovation Institute, Abu Dhabi, UAE
[d]KUCARS, Department of Electrical Engineering and Computer Sciences, Khalifa University, UAE
[e]University of Florence, Florence, Italy
[f]University of Science, Ho Chi Minh city, Vietnam
[g]Vietnam National University, Ho Chi Minh city, Vietnam
[h]Center for Mathematical Morphology - Mines ParisTech - PSL France
[i]Department of Engineering, Pontifical Catholic University of Peru, PUCP Peru
[j]Millennium Institute Foundational Research on Data, Department of Computer Science, University of Chile, Chile
[k]Electrical and Computer Engineering Department, University of Patras, Rion-Patras, Greece
[l]Department of Computer Science Aberystwyth University, Aberystwyth, SY23 3DB, UK
[m]Department of Computer Science Edge Hill University, Ormskirk, L39 4QP, UK
[n]Centre de Recherche et Restauration des Musees de France, Paris, France
[p]Université Bourgogne Franche-Comté, Dijon, France
[o]Norwegian University of Science and Technology, Gjovik, Norway

August 17th 2020

Abstract

This paper presents the methods that have participated in the SHREC'20 contest on retrieval of surface patches with similar geometric reliefs and

the analysis of their performance over the benchmark created for this challenge. The goal of the context is to verify the possibility of retrieving 3D models only based on the reliefs that are present on their surface and to compare methods that are suitable for this task. This problem is related to many real world applications, such as the classification of cultural heritage goods or the analysis of different materials. To address this challenge, it is necessary to characterize the local "geometric pattern" information, possibly forgetting model size and bending. Seven groups participated in this contest and twenty runs were submitted for evaluation. The performances of the methods reveal that good results are achieved with a number of techniques that use different approaches.

1 Introduction

Figure 1: A visual representation of the challenge proposed in this contest. A query model Q with a bark-like relief impressed on its surface is selected. In the ideal case, models with a bark-like relief are retrieved before than models with different reliefs, independently of the global geometry of the models. The "check" and "cross" marks highlight models that are relevant or non-relevant to the query.

Geometric reliefs are a significant component for the local characterization of a surface, which are independent of its overall shape and spatial embedding. Being able to characterize different repeated relief patterns on a surface is a key issue for several tasks, such as the analysis and detection of molding marks, composite materials and ornamental decorations on an object surface. The characterization of this local surface property is an open problem that is gaining more and more interest over the years.

Several methods have been introduced for the characterization of local, repeated, geometric variations on a surface, showing this is a vivid research field. In the set of methods that face this problem, we distinguish two main strategies: i) to (fully or partially) project a 3D model into an image or a set of images and then apply a texture image retrieval method; ii) to extend the image texture characterization directly to 3D model representations. Examples of methods that face this problem as an image texture retrieval problem have been proposed for the classification of trees based on their bark reliefs [34] or the classification of engraved rock artifacts based on their height-fields [52]. In this trend, the combination of the SIFT descriptor with the Fisher Vector, which gave very good performances for image texture retrieval [16], has shown very

good performances also for the retrieval of relief patterns [19]: in this case, representative surface images were obtained by projecting the mean curvature of the neighborhood of the center of the model. Methods that directly use 3D representations are generally designed for triangle meshes or point clouds, because these representations allow a precise and locally adaptive representation of the surface that is less accessible with grids (e.g., voxels). This class of methods includes the numerous extensions of the the Local Binary Pattern (LBP) [32] proposed in recent years. The first of these extensions was the meshLBP [48, 47], followed by the edgeLBP [26, 27, 28, 11, 31] and the mpLBP [29, 30]. Besides the different strategies to encode the neighbour of a vertex, the main idea behind these LBP-based 3D characterizations is to replace the gray-scale value in the pixels of an image with geometric or colorimetric properties (e.g., curvatures or color channels) defined on the faces or the vertices of the model. Recently, also the multi-scale properties of the Laplacian operator have been used in [33] to obtain a scale-aware surface description. In this case, the parts of interest are obtained by analyzing the difference between a surface and its counterpart obtained by smoothing.

Based on the increasing number of methods for 3D pattern retrieval made available in recent years, we think it is now important to understand how much existing methods are suitable to address realistic applications. The aim of this SHREC 2020 track is to provide a new benchmark for geometric pattern retrieval and to evaluate methods for assessing the similarity between two objects, only on the basis of the local, geometric variations of their surfaces, without considering their global shape. Our new collection of 3D models is characterized by different classes of reliefs on the models surface. A visual representation of the task addressed in this contest is shown in Figure 1.

These reliefs represent different kinds of materials, like bark wood or rocks, and structures, like bricks. The peculiarity of the models proposed in this contest is that a realistic geometric pattern (derived from real texture images) is applied to a number of base models, some of those have a non-trivial topology (with handles, tunnels, boundaries, etc.).

The remainder of the paper is organized as follows. Section 2 briefly overviews existing datasets and benchmarks that address the geometric pattern retrieval or strictly related tasks. Section 3 describes the 3D models used in this challenge and details how they have been generated from a base model and a set of real textures. Section 4 details the eight methods submitted to this contest, while Section 5 introduces the methodology and the measures used to evaluate the different runs. Section 6 presents the settings of the runs submitted to this contest and their retrieval and classification performances. Finally, discussions and concluding remarks are in Section 7.

2 Related benchmarks

The interest for geometric pattern analysis has been borrowed from image texture analysis, which is a typical problem of Computer Vision. To the best of

our knowledge, the first dataset explicitly delivered for 3D texture analysis was the "MIT CSAIL Textured Models Database" [1]. During years, several factors have concurred to the increase of collections of 3D models equipped with textures; for instance, the improvement of the spatial data acquisition systems that also allow the representation of the surface details; the increase of applications interested in the comparison of 3D models on the basis of their texture or material and, even, the success of benchmarks and methods for image texture retrieval [16].

Without aiming to list all the existing general purpose data collections that contain some model equipped with a 3D texture (e.g., Skechfab [3] or Turbosquid [5]), we focus on benchmarks for similarity evaluation that provides also a ground truth and a number of evaluation measures. Several previous SHape REtrieval Contest (SHREC) tracks are somehow related to our challenge. The first SHREC track that partially faced the problem of local surface characterization is the SHREC'13 track on retrieval and classification of 3D textured models [13], extended the SHREC'14 track [9] with the same task but a larger dataset. A complete analysis of the methods tested on those contests was published in [10]. Differently from this contest that only focuses on local, geometric surface variations, there, the task was to group models based on their overall shape and their colorimetric texture. In other words, models that were globally similar but with different textures were less similar than those with the same shape and texture. While in the SHREC'13 and SHREC'14 tracks, texture analysis was colorimetric and only marginal, here it is geometric and the only aspect that drives the similarity among models.

The interest on geometric reliefs shaped into the SHREC'17 track on the retrieval of reliefs [11]. There, fabrics with different patterns were acquired with photogrammetry and used to create a benchmark for the pattern retrieval task. That benchmark entirely focused on the local characterization of surfaces based on patterns and it is currently used as the reference benchmark by many of the works on this topic. The high number of subscribers to that track, but the quite limited number of effective runs submitted to the track revealed the high interest in the subject and the difficulty in facing that task. Aside from some highlights, the methods submitted to the original contest showed quite limited performances, later on the research on this topic progressed, several methods have been proposed and successfully tackled such a benchmark. On a similar note, the SHREC'18 track on gray patterns [31] proposed a retrieval task on a dataset of models characterized by gray-scale patterns. Interestingly, all the participants proposed feature-vector based methods.

It is also worth mentioning the SHREC'18 track on geometric pattern recognition [12] that differs from the previous benchmarks on 3D pattern retrieval because the participants were asked to locate a query relief sample in a set of 3D models. The challenge was to recognize if a type of geometric pattern is contained or not in another model and, eventually, to identify it on the model. The challenge launched in that task is still an open problem and that track report can be considered as a position paper on 3D pattern recognition.

Based on the progresses made in recent years, it is now important to analyse

how much the performance of the various approaches has improved compared to the methods presented in the SHREC'17 track [11], while bearing in mind that the more general issue of 3D pattern recognition presented in the SHREC'18 track [12] is still open.

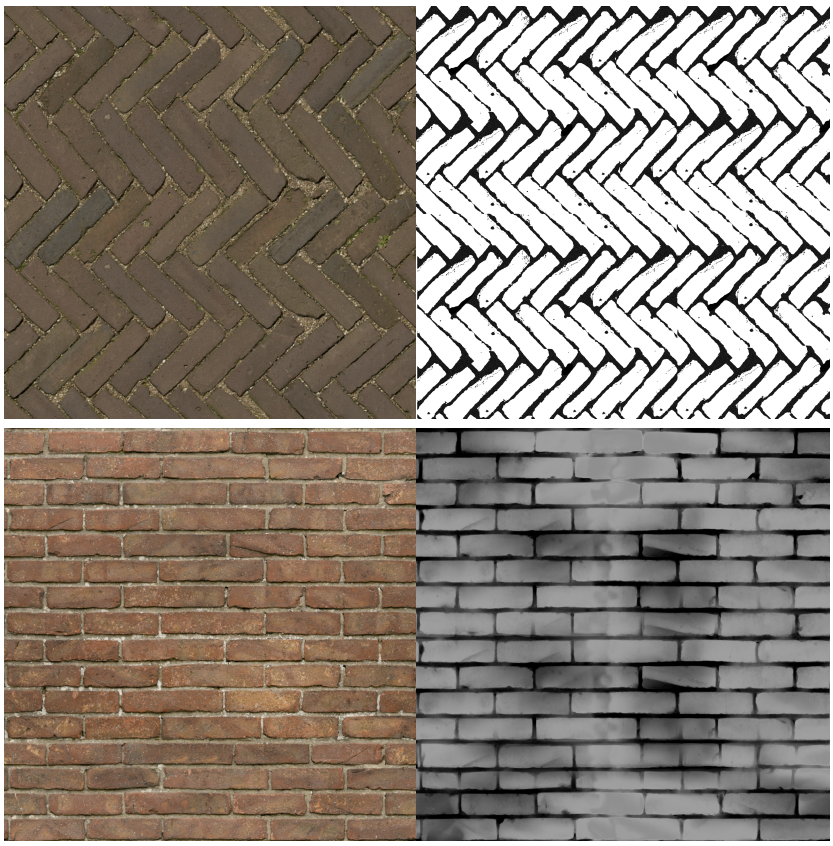


Figure 2: An example of the transformation process from texture to height map. On the left, the original textures are shown. On the right, the final height-map obtained with the process explained in Section 3. This process can end with a binary image (just black and white, as in the example at the Top) or a gray-scale one (like that at the Bottom).

3 The dataset

The dataset proposed for this challenge consists of 220 triangulated surfaces. Each one of them is characterized by one of 11 different geometric reliefs.

To create the models, we selected the 20 base models already used in [31]. These models represent pots, goblets and mugs. The surfaces of these models

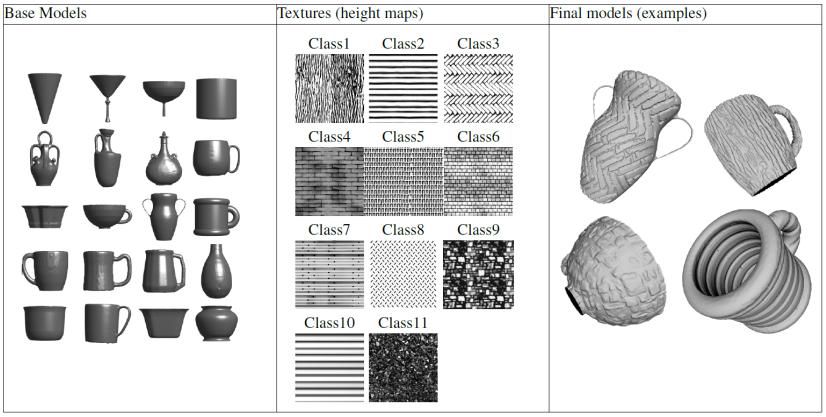


Figure 3: (Left): the 20 base models on which the reliefs are applied. (Center): the 11 transformed textures used as height-fields on the base models (the brighter the color, the higher is the value of the field in that point). (Right): a sample of the final models of the dataset of the contest.

are properly oriented and they are made of a single connected component. The topology of some models is non trivial (they may contain handles or tunnels) and may present a boundary, depending on the object represented. Then, a set of 11 textures is selected from the free dataset of textures available online from the site Texture Haven [4] that contains a set of natural, high quality texture images made from scanned maps. Most of these textures represent real bricks, floors, roofs surfaces and rock or wood materials.

Given the nature of the textures selected, on the one hand, models of buildings or their agglomerates would be the most realistic; on the other hand, we think that in 3D pattern retrieval the most challenging issue is to deal with free-form models, possibly with more complex bendings and non trivial topology. Given the heterogeneity of the textures selected and the geometry of the base models, methods that perform well in this contest have a high chance of being equally valid in other contexts, with little to no changes.

We transform each texture in height values suitable to create a geometric relief by converting each texture into a gray-scale image. The brightness and the contrast values of each image were tuned for each image, based on the values that better enhance the details of the respective color texture. The obtained height field map is applied to the models: initially, the texture is projected onto the target model. Depending on the surface bending, this procedure deforms the texture. To limit this effect, each model is fixed by hand, in particular, in correspondence of significant distortions and parts of the surface with complex geometry (like tight handles). Finally, we rise the vertices of the triangle mesh based on the gray-scale value of the previously processed image along the normal vectors of the models. The same process is repeated for all the textures. A couple of examples of the conversion of a texture into a height map are depicted

in Figure 2.

Finally, the models are slightly smoothed to minimize the perturbations in the color derived from the gray-scale conversion of the textures and the models are sampled with 50000 vertices. Base models, height fields and examples of the final 3D models are shown in Figure 3.

The Ground Truth

The challenge proposed in this contest is to group the models only according to the geometric reliefs impressed on them, rather than their shape. In other words, a perfect score is obtained if a method is able to define 11 groups of 20 models each, each group with the models characterized by one of the 11 different geometric reliefs.

4 The participants and the proposed methods

Seven groups subscribed to this track. All of them submitted at least one method; one group submitted two methods; overall, eight methods and twenty runs were submitted to evaluation. The participants are anonymous for review for and their proposed method(s) are summarized in the following.

4.1 Augmented Point Pair Feature Descriptor Aggregation with Fisher Kernel (APPFD-FK) by Ekpo Otu, Reyer Zwiggelaar, David Hunter, Yonghuai Liu

The Augmented Point Pair Feature Descriptor (APPFD) is a 3D object descriptor made of local features that capture the geometric characteristics or properties of a set of surface patches, each centred at a point (i.e. a *keypoint*) $p_{k_i} = [x, y, z]$, which incorporates the geometrical relation between p_{k_i} and its r-nearest neighbors (i.e. the surface patch around p_{k_i}). The APPFD algorithm consists of the following stages: point cloud sampling, surface normals estimation, keypoints determination, local surface patch (LSP) selection, Augmented Point-pair Features (APPF) extraction and keypoints descriptor (APPFD) computation for LSPs. While the APPF extraction and APPFD algorithms are described in detail here, the reader is referred to the literature in [35] for more details on the other stages. Finally, the Fisher Kernel approach to local descriptor aggregation with Gaussian Mixture Model (GMM) [22, 39] is applied to the local APPFD to derive a single signature, APPFD-FK, for each 3D shape. Figure 5 shows the processing pipeline of the APPFD-FK algorithm. The three main steps of the algorithm are outlined in the following:

1. *Augmented Point Pair Feature Descriptor (APPFD):* The APPFD is derived by three sub-steps: ***i)*** For each LSP extracting four-dimensional local Point-Pair Feature (PPF), $f_1 = (\alpha, \beta, \gamma, \delta)$ as in [46], ***ii)*** Augmenting f_1 to a six-dimensional feature - the Augmented PPF, using

additional two-dimensional local angular feature, $f_2 = (\theta, \phi)$, depicted in Figure 4, and ***iii)*** Discretizing the six-dimensional augmented feature $f_3 = (\alpha, \beta, \gamma, \delta, \theta, \phi)$ into one or multi-dimensional histograms to yield the final local APPFD. Firstly, extracting PPF involves two sets of *oriented* points, $p_i, p_j = [(p_i, n_i), (p_j, n_j)]$, used to encode the underlying surface geometry for their patch on a 3D surface. For every possible combination q of p_i, p_j in LSP (i.e. r-neighbourhood of p_{k_i}), where p_i is the **source point** *w.r.t.* the constraint in (1) holding TRUE, where $i \neq j$, then a transformation independent Darboux frame, $D_f = U, V, W$ is defined as: $U = n_i$, $V = U \times ((p_j - p_i)/\delta)$, $W = U \times V$.

$$|n_i \cdot (p_j - p_i)| \leq |n_j \cdot (p_j - p_i)| \tag{1}$$

Alternatively, p_j becomes the **source point** (i.e. point with the larger angle between its associated normal and the line connecting the two points) if the constraint in (1) is FALSE, and the variables in (1) are reversed. $f_1(p_i, p_j) = (\alpha, \beta, \gamma, \delta)$ is then derived for the source point as follows:

$$\alpha = \arctan(W \cdot n_j, U \cdot n_j), \;\; \alpha \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \tag{2}$$

$$\beta = V \cdot n_j, \;\; \beta \in (-1, 1) \tag{3}$$

$$\gamma = U \cdot \frac{p_j - p_i}{\|p_j - p_i\|}, \;\; \gamma \in (-1, 1) \tag{4}$$

$$\delta = \|p_j - p_i\|. \tag{5}$$

Secondly, $f_2(p_i, p_j) = (\theta, \phi)$ is extracted for every possible combination of point-pair, p_i, p_j in the LSP, because f_1 is not robust enough to capture the entire geometric information for a given surface region or LSP. In addition, the PPF approach opens up possibilities for additional feature space. Therefore, as illustrated in Figure 4, θ is geometrically the angle of the projection of the vector, $\overrightarrow{S}$ onto the unit vector $\overrightarrow{V_1}$, and ϕ is the angle of the projection of the vector $\overrightarrow{S}$ onto the unit vector $\overrightarrow{V_2}$, where $\overrightarrow{V_1} = p_i - p_c$, $\overrightarrow{V_2} = p_i - l$, and $\overrightarrow{S} = p_i - p_j$, with $p_c = \frac{1}{n_i}\sum_{i=1}^{n_i} p_{k_i}$ (i.e. LSP centroid), and $l = (p_j - p_c)$, the vector location of p_{k_i} *w.r.t.* its LSP. Note that p_i, p_j, p_c, and l are all points in $\mathbb{R}^3$ space, although l is a vector.

Basically, α, β, γ are the angular variations between (n_i, n_j), while δ is the spatial distance between p_i and p_j. In Euclidean geometry, each of the projections ϕ and θ can be interpreted as an angle between two vectors. For example $\angle_1\langle \overrightarrow{S}, \overrightarrow{V_1}\rangle$ and $\angle_2\langle \overrightarrow{S}, \overrightarrow{V_2}\rangle$ are equivalent to θ and ϕ respectively. These angles are derived by taking the scalar products of $(\overrightarrow{S} \cdot \overrightarrow{V_1})$ for $\angle_1$, and $(\overrightarrow{S} \cdot \overrightarrow{V_2})$ for $\angle_2$ about a point p_i in a given LSP. Mathematically, scalar products defined in this manner are homogeneous (i.e. invariant) under scaling [50] and rotation [24]. For this reason, our two-dimensional local geometric features, θ, ϕ are rotation and scale invariant

for 3D shapes under rigid and non-rigid affine transformations. Moreover, notice that since geometric information are embodied by these variations and projections, the global shape of the 3D shape is not considered at all.

Lastly, for each LSP or keypoint, p_{k_i} with q combinations, $q(q-1)/2$ six-dimensional APPF: $f_3 = (f_1 + f_2)$ is obtained thus: $f_3(p_i, p_j) = (f_1(p_i, p_j), f_2(p_i, p_j)) = (\alpha, \beta, \gamma, \delta, \phi, \theta)$ and descritized into histograms to yield APPFD. In computing APPFD for this task, 4500 points and their normals, (P, N) were sampled from each 3D surface, K keypoints, $\{p_{k_i}, i = 1 : K\}$ were selected and for each p_{k_i}, a LSP, $\{P_i, i = 1 : K\}$ and their corresponding normals, $\{N_i, i = 1 : K\}$ were computed. Points in P_i are within the specified radius, $r = 0.30 - 0.40$ around p_{k_i}.

For our first and second experimental runs (APPFD-FK-run1 and APPFD-FK-run2) a one-dimensional $[0, 1]$ *normalized* histogram with bins = 35 is used to represent each of the feature-dimension in APPF, concatenated to yield a final 6 times 35 = 210-dimensional local descriptor for each LSP or keypoint. In our third experimental run (APPFD-FK-run3) all six-dimensional feature in APPF are discretized into a multi-dimensional histogram with bins = 5 in each feature-dimension, flattened and normalized to give $5^6 = 15625$-dimensional local descriptor for each LSP or keypoint.

2. *Keypoint APPFD Aggregation with Fisher Vector (FV) and Gaussian Mixture Model(GMM):* Inspired by the work in [22, 39], the final stage of our novel APPFD-FK algorithm consists of computing a global FV for each input 3D shape given their keypoint APPFDs. The FV computation relies on training a GMM, as a generative probabilistic model, with the keypoint APPFDs for all database shapes. The GMM is trained with 10 Gaussians, using diagonal covariances for all experimental runs. Using the trained GMM and local keypoint APPFDs for a given 3D shape, a final global FV which is L_2 and power-normalized (so it has unit length) is computed with the help of [2]. Then, for local APPFD with 210 and 15625 dimensions, FVs with 4210 and 312510 dimensions, respectively are returned, which represent a single 3D shape. However through experimental findings, applying linear dimensionality reduction (in our case principal component analysis, PCA) to either of the 4210 or 312510 dimensional FVs remaining 99% of their information reduces them to 162 or 186 respectively, and still yield close matching results.

3. *Shape Similarity Measurement:* Overall, the L_2 or cosine distance metric between FVs is expected to give a good approximation of the similarity between shapes in the sc20-relief-rc dataset. The cosine metric in (6) is

adopted, instead.

$$\begin{aligned}\cos(\mathbf{FV_1}, \mathbf{FV_2}) &= \frac{\mathbf{FV_1} \cdot \mathbf{FV_2}}{\|\mathbf{FV_1}\| \quad \|\mathbf{FV_2}\|} \\ &= \frac{\sum_{i=1}^{n} \mathbf{FV_1}_i \mathbf{FV_2}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{FV_1}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{FV_2}_i)^2}}\end{aligned} \tag{6}$$

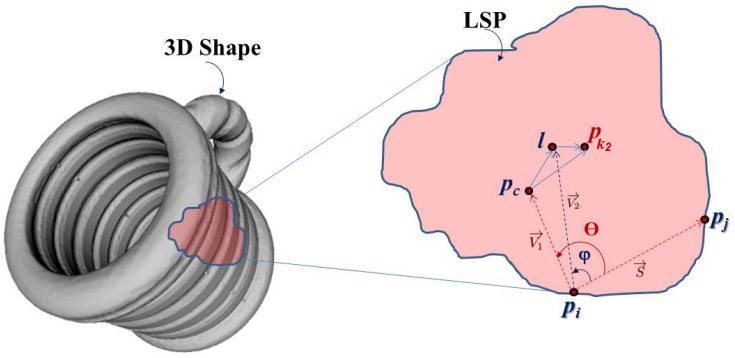


Figure 4: Local Surface Patch (LSP), P_i with pairwise points (p_i, p_j) as part of a surflet-pair relation for (p_i, n_i) and (p_j, n_j), with p_i being the origin. θ and ϕ are the angles of vectors projection about the origin, p_i. θ is the projection angle from vector $\langle p_i - p_j \rangle$ to vector $\langle p_i - p_c \rangle$ while ϕ is the projection angle from vector $\langle p_i - p_j \rangle$ to vector $\langle p_i - l \rangle$. The LSP centre is given by p_c, keypoint is given as p_{k_i} where $i = 2$. Finally, l is the vector position of $p_{k_i} - p_c$.

4.2 Orientation Histogram (OH) and Deep Feature Ensemble (DFE) by Hoang-Phuc Nguyen-Dinh, Minh-Quan Le, Hai-Dang Nguyen and Minh-Triet Tran

This group submitted two different methods, with three runs each. Since the two methods share the pre-processing steps, we describe both methods in this Section. As the goal of the track is to retrieve 3D models based only on the relief of their surfaces and not the shape of the 3D models, the authors do not exploit the 3D mesh directly but take the 2D screenshots of the 3D models. Best view among multiple 2D screenshots is selected by searching the maximum inscribed rectangle.

4.2.1 Orientation Histogram (OH)

1. *Uprighting and rendering a 3D object:* The first step is to upright the 3D object by transforming the object into a new 3D coordinate system so that the object stands vertically across the y-axis for the ease of rendering.

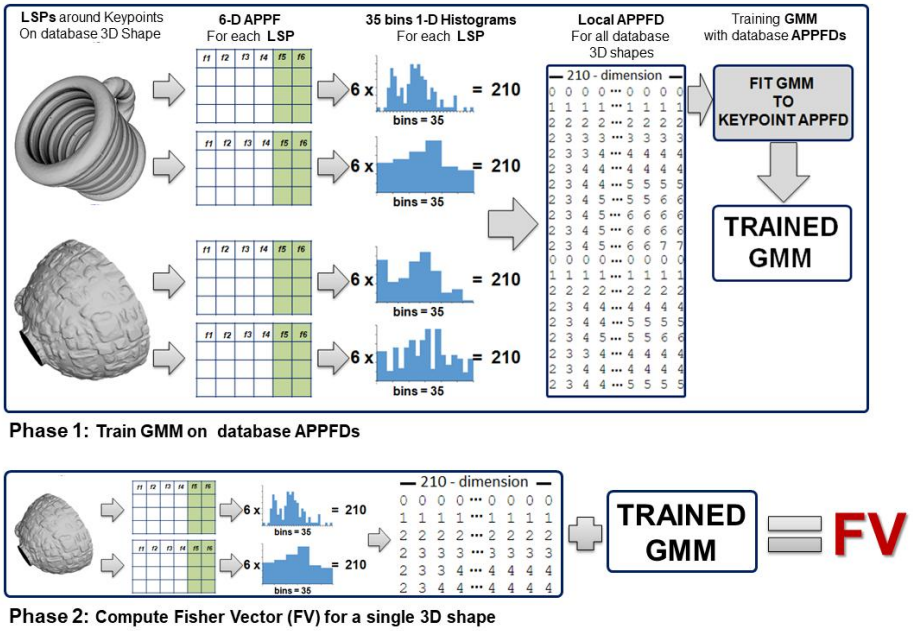


Figure 5: Overview of the APPFD-FK framework, which computes a global Fisher Vector (FV) for each 3D shape.

That could be done by finding the eigenvector of all the vertices of the object and then choose the normalized version of it (called vector j') to be the O'_y axis of the new system. The two remaining axes O'_x and O'_z are chosen randomly, satisfying that all the three vectors are unit vectors and pairwise orthogonal. The origin of the new system is the centroid of the object. Moving the camera around the O'_y axis, many 2D images of the object are sampled. Among these, the image having the most relief patterns is selected. As plain images would have fewer points at which the gradient vectors equal to zero and vice versa, the Sobel Filter [23] is used to calculate the gradients of an object's rendered images. The image with the most non-zero gradient vectors is set to be the one representative of this object. An overview of this step is shown in Figure 6.

2. *Finding the largest inscribed square:* In order to remove the global shape of the object and focus on the local reliefs, the largest inscribed square of the object on the 2D image is extracted, which means selecting the largest region of only relief patterns. This is done by solving the problem of finding the largest square with no white points inside of it (because white points are background). The latter problem is resolved by using a simple binary search algorithm with the complexity of $N^2 * \log N$ (N is the greater value between the width and the height of an image). After

this step, each 3D object has one representing a 2D square image.

3. *Feature extraction:* The goal of this step is to represent every image after the second step as feature vectors with the length of N. Such vector is obtained with the method of counting the "gradient histogram" of an image. Specifically, in each image, first, Sobel Filter [23] is used to find the gradient vectors of every point and derive their modules and their angles with the O_x axis. Second, a histogram with the number of bins being N, ranging from $-\pi/2$ to $\pi/2$, is computed on the frequency of the calculated angles. Every angle is counted with the weight of its corresponding vector's module instead of one as usual. Furthermore, the weight of a sample is distributed to the two nearest bins with a suitable ratio instead of just one. This histogram could describe the direction and size of the relief on an image. The histogram is then normalized by making the sum across N bins be 1 and translating the histogram so that the highest bin is the first bin (ranged from $-\pi/2$ to $\pi/2 + \pi/n$). Every histogram is saved as a 1D-array called the feature vector of the image.

4. *Creation of the Dis-similarity Matrix:* The distance between pairs of 3D objects is calculated on their feature vectors using suitable metrics, such as L1 distance, L2 distance, chi-square distance, cosine-distance, etc.. The original distance matrix is then created by calculating the distance between every pair of vectors. Authors aim to further exploit the visual relationships of an object x and its neighbors with another object y. Therefore, the authors use the Average Query Expansion (AQE)[6] to modify the original distance matrix (see Figure 7. Let $R(x)$ be the list of the nearest neighbors (in the ascending order) of the object x. The modified distance between object x and object y is defined as follows:

$$dist_{AQE}(x, y) = \alpha \times dist(x, y) + \frac{1-\alpha}{k} \times \sum_{i=1}^{k} dist(R(x)_i, y)$$

where dist is the original distance, dist_AQE is the modified distance matrix, k and α are hyperparameters.

An overview of these last two steps is shown in Figure 8. The runs submitted to the track differ for the number of bins and the metrics for the feature vectors, their settings are described in Section 6.1.

4.2.2 Deep Feature Ensemble (DFE)

This method shares the first two steps (i.e., the model pre-processing steps) with the method described in Section 4.2.1. The third and fourth steps are described in the following.

3. *Use of pre-trained models to extract features:* With the advances of deep learning, especially pre-trained Convolutional Neural Networks (CNNs),

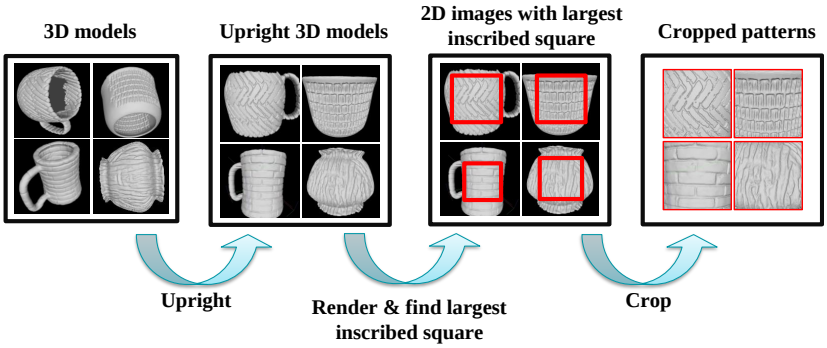


Figure 6: Overview of the pre-processing steps for the OH and DFE methods.

the authors propose using these pre-trained models to extract features of each pattern. Many high-performance models such as ResNet [20], DenseNet [21], VGG [40], and Efficient-Net [43] suit this purpose.

A common approach is to use an extracted feature vector from a pre-trained network as the input for classification. However, the output at each layer in a pre-trained model offers different high-level information about the textures in the original input. Therefore, the authors propose to synergize the information extracted from different intermediate layers of different pre-trained networks by assembling feature vectors.

The authors choose intermediate layers instead of the last ones because features extracted in the middle layers would be more appropriate to represent information of the simple patterns on the texture input image. First, the authors pass a square image containing patterns into a pre-trained neural network. Then, the authors take the output tensor of a chosen intermediate layer of that network with the shape of (h, w, $channelsize$). After that, the authors pass the tensor through a Global Average Pooling Layer to create a vector with a length of ($channelsize$,) used as a feature vector. By using Global Average Pooling, the authors pick up all requisite activated features without missing any of them as using Global Max Pooling and make the result more robust to spatial translation in the image. Finally, the authors multiply each feature vector by a parameter (see Section 6.1) and concatenate them into one single final feature vector. A visual representation of the way authors ensemble the feature vectors from different layers in different models is shown in Figure 9

4. *Creation of the Dis-similarity Matrix:* After extracting features by the method described above, each object is represented as a feature vector. Such vectors are used to calculate distance between all pairs of objects (with metrics such as cosine similarity, L1 distance, L2 distance, etc.). Besides, the authors combine Average Query Expansion (AQE)[6] with a view to helping our model to remove noises.

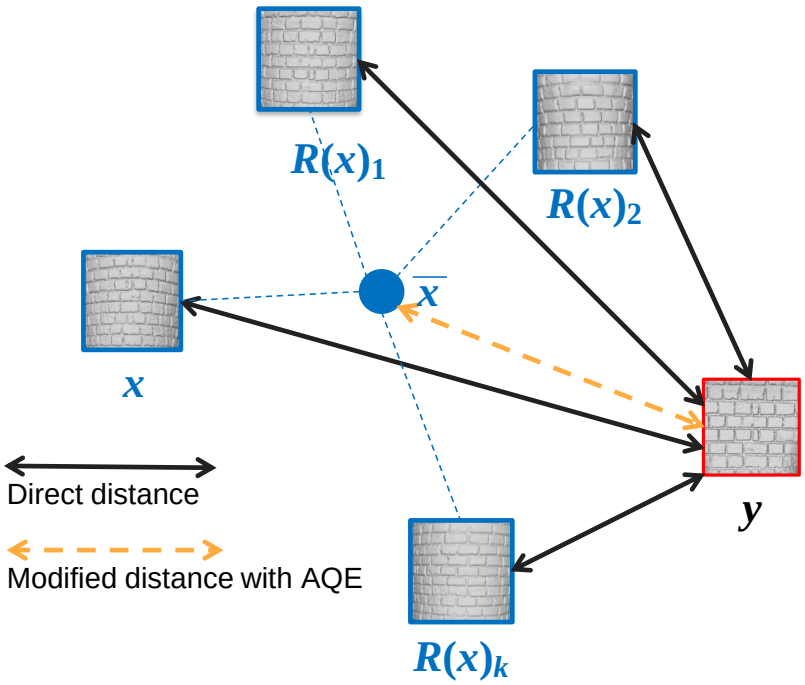


Figure 7: Overview of the Average Query Expansion used in OH and DFE.

The pipeline of DFE method is summarized in Figure 10. The authors considered many single pre-trained models.; the pre-trained models considered in the runs submitted to this track are described in Section 6.1.

4.3 Deep Patch Metric Learning (DPML) by Leonardo Gigli, Santiago Velasco-Forero, Beatriz Marcotegui

This method works in two main steps. The first one involves the extraction of patch images from the mesh surfaces, to decorrelate information about relief from the global shape of the mesh. The second step uses these patches to train a Siamese Neural Network [15] to learn a distance function between each pair of images.

1. *Patch extraction:* The goal is to extract images containing only the local texture. Let us define a triangle mesh $\mathcal{S} \subset \mathbb{R}^3$, along with a graph $\mathcal{G}_\mathcal{S} = (\mathcal{V}, \mathcal{E})$ associated to $\mathcal{S}$, that is the graph whose nodes are the points $(x_1, \ldots, x_n)$ of $\mathcal{S}$. Two nodes are connected if and only if they are vertices of one of the triangles of $\mathcal{S}$ at the same time. With this setting, the method starts sampling a subset of points $(x_1, \ldots, x_n) \in \mathcal{S}$ using Poisson Disk Sampling [51]. Then for each point x_i, using the geodesic distance defined over the graph $\mathcal{G}_\mathcal{S}$, a local neighborhood is defined. In particular,

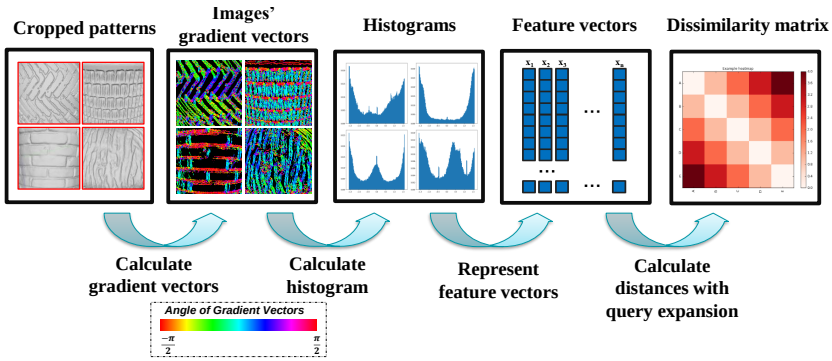


Figure 8: Overview of the third and fourth steps of the OH method.

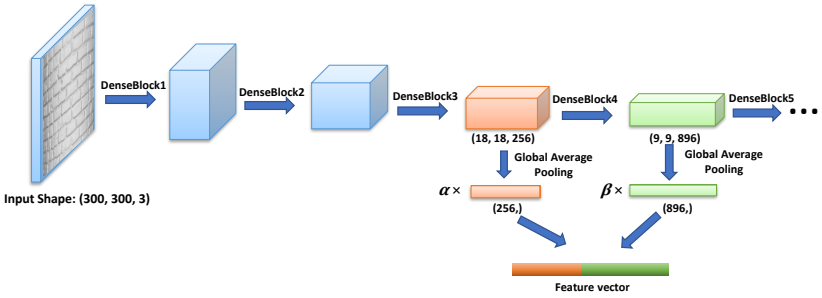


Figure 9: Illustration for step three "Extracting Feature by concatenating feature vectors from different layers of pre-trained models" of the DFE method (the figure illustrates the step when using Dense-Net-201).

the geodesic distance d between two points x_i and x_j is the length of the shortest path connecting them. Thus, given $r > 0$, the local neighborhood is defined as $\mathcal{N}_r(x_i) = \{x_j \in \mathcal{V} | d(x_i, x_j) \leq r\}$. The goal is to project the local neighborhood over a plane and obtain an elevation image. For this reason, only the neighborhood that are as flat as possible are selected. To estimate such a property, the covariance based features are used. Those features are derived from the eigenvalues $\lambda_1 \leq \lambda_2 \leq \lambda_3$ of the neighborhood covariance matrix defined as:

$$cov(\mathcal{N}_r(x_i)) = \frac{1}{|\mathcal{N}_r(x_i)|} \sum_{x \in \mathcal{N}_r(x_i)} (x - \overline{x})(x - \overline{x})^T$$

and $\overline{x}$ is the centroid of the neighborhood $\mathcal{N}_r(x_i)$. The following criteria are used to estimate if the neighborhood is flat enough:

- criterion on planarity: $\frac{\lambda_2 - \lambda_3}{\lambda_1} \geq 0.5$,
- criterion on the change of curvature: $\frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \leq 0.03$.

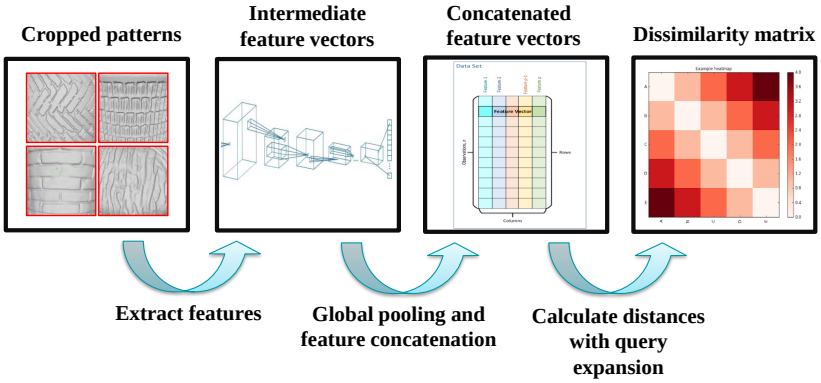


Figure 10: Overview of the third and fourth steps of the Deep Feature Ensemble method (DFE).

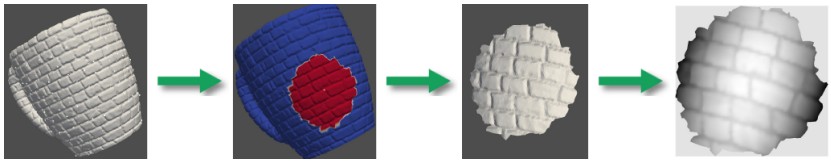


Figure 11: Pipeline of the first step of the DPML method.

The two values have been chosen empirically after some test over different objects. Validated neighborhoods are projected over the tangent space of the surface at x_i. A regular grid is defined over the tangent space, and each element of the grid corresponds to a pixel of the image. The intensity values of the image correspond to the distance between the points projected over the element and the tangent plane. In order to obtain a uniform sized patch the method crops them to obtain images of size 231×231 (equal to the smallest image extracted with this process). For each patch, crops are computed so that there is the minimum number of void pixels in each image. An overview of this process is reported in Figure 11.

2. *Learning the embedding:* The selected images are used to train a Siamese neural network with the Triplet Loss. The architecture is composed of three CNNs sharing the same weights. In this case the VGG16 [41], without fully connected layers, is chosen as CNN. The CNNs work in parallel taking as input a triplet of images and generating a comparable feature vectors, as shown in Figure 12. The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity, i.e. an image from a different object [14].

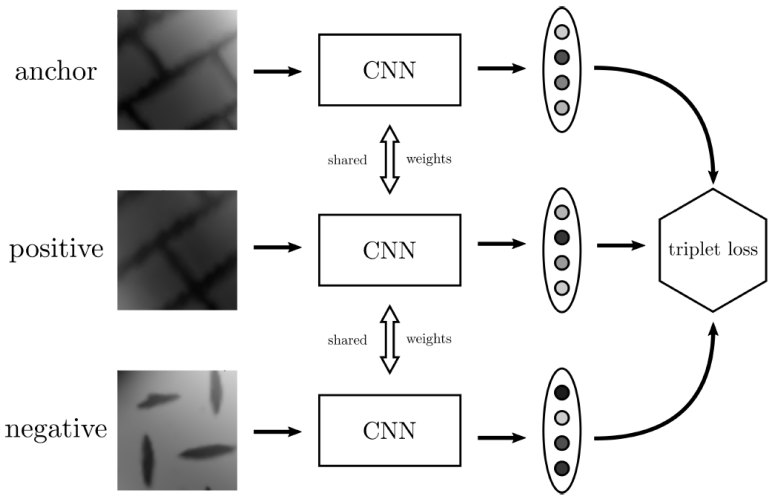


Figure 12: Overview of the network of the DPML. Such network consists of a batch input layer and a deep CNN which results in the image embedding by using a triplet loss during training.

Finally, the distance Δ between two objects $\mathcal{S}_i$ and $\mathcal{S}_j$ is defined as the minimum distance between any couple of images belonging to the two surfaces:

$$\Delta(\mathcal{S}_i, \mathcal{S}_j) = \min_{h,k \in \{1,\ldots,m_i\} \times \{1,\ldots,m_j\}} \delta(I_h, I_k),$$

where $\delta(I_h, I_k)$ is the similarity function learned by the Siamese neural network.

The authors submitted two runs for this method. The different parameter settings are reported in Section 6.1.

4.4 Signature Quadratic Form Distance and PointNet (PointNet+SQFD) by Ivan Sipiran and Benjamin Bustos

This method consists of computing the distance between two shapes using the Signature Quadratic Form Distance [8] (SQFD) over descriptions of local patches. The SQFD distance has proven to be effective in large-scale retrieval problems where shapes are represented as sets of features [42]. This approach focuses the attention in the relief (instead of the entire shape) by decomposing the shape into local patches and describing the local patches using a neural network. Subsequently, authors compute aggregated features that keep the local variability of the patches. Finally, the SQFD distance is used to compare two signature collections.

Given the 3D shape M, the feature set F_M contains descriptors for the shape. To use the SQFD distance, the feature set F_M has to be clustered in

a set of disjoint descriptors, such that $F_M = C_1 \cup C_2 \cup \ldots C_n$. A signature is computed for each cluster, defined as $S^M = \{(c_i^M, w_i^M), i = 1, \ldots, n\}$, where $c_i^M = \frac{\sum_{d \in C_i} d}{|C_i|}$ and $w_i^M = \frac{|C_i|}{|F_M|}$. Each signature contains the average descriptor in the corresponding cluster and a weight that quantifies the representative power of the cluster with respect to the entire feature set. The clustering method uses an intra-cluster threshold (λ), and an inter-cluster threshold (β) and a minimum number of elements per cluster (Nm) to perform the grouping. For more details about the local clustering method, see [42].

Given two objects M and N, and their corresponding signatures S^M and S^N, the SQFD distance is obtained as follows:

$$SQFD(S^M, S^N) = \sqrt{(w^M| - w^N) \cdot A_{sim} \cdot (w^M| - w^N)^T},$$

where $(w^M|w^N)$ denotes the concatenation of weight vectors. Matrix A_{sim} stores the correlation between averaged descriptors in the signature. The correlation coefficient between two descriptors is defined as:

$$corr(c_i, c_j) = exp(-\alpha d^2(c_i, c_j)).$$

Given an input shape, p local patches of diameter *diam* are sampled. The first seed vertex is randomly selected from the shape, while the remaining vertices are chosen using a farthest point sampling strategy over geodesic distances. For each selected vertex, a local patch of diameter *diam* is computed, using a region growing method. Each local patch is used to obtain a point cloud that represents the patch. In all the submitted runs, a local patch is sampled with 2500 points. For the description of a given point cloud, a PointNet neural network [36] is used. A PointNet network using the ModelNet-10 dataset [53] is pre-trained for the classification task. After training, the neural network is fed with the point clouds obtained from the previous procedure. The 1024-dimensional feature obtained by PointNet is used before the classification of the network.

In the end, each shape in the dataset is represented by p descriptors of 1024 dimensions, which are finally used to compute the signatures and the SQFD. The other parameters that characterize the runs are the number and diameter of the patches. More details on these settings are reported in Section 6.1.

4.5 Smooth-Rugged Normal Angle (SRNA) by Ioannis Romanelis, Vlassis Fotis, Gerasimos Arvanitis, Konstantinos Moustakas

This approach outlines the geometric texture by using a per-vertex quantity and extracts a representative feature vector which is used to test against every other model in the database.

Consider a smooth planar surface on which a transformation matrix $\mathbf{T_1}(\mathbf{v_i})$ is applied on each of its vertices, "bending" it in such a way, that it forms a smooth cylinder (see Figure 11(b, left)). As is to be expected, $\mathbf{T_1}(\mathbf{v_i})$ is

different depending on $\mathbf{v_i}$. By applying a pattern on the planar surface, while retaining one-to-one correspondence between the vertices, the surface of Figure 11(b, center) is obtained. The new vertices will have moved by some distance $\epsilon_\mathbf{i}$ from their original positions yielding $\hat{\mathbf{v}}_\mathbf{i} = \mathbf{v_i} + \epsilon_\mathbf{i}$. This surface can also be morphed into a cylinder, using a transformation $\mathbf{T_2}(\hat{\mathbf{v}}_\mathbf{i})$. With sufficient vertex density it is possible to state that $\mathbf{T_1}(\mathbf{v_i}) \approx \mathbf{T_2}(\hat{\mathbf{v}}_\mathbf{i})$.

Since the transformation matrices affect not only the shape's vertices, but also its vertex normals, we can conclude that the angle between the normal vectors$\mathbf{n_i}$ (smooth) and $\hat{\mathbf{n}}_\mathbf{i}$ (with pattern) is preserved on both the plane and the cylinder. This implies that the quantity $\theta_i = \angle(\mathbf{n_i}, \hat{\mathbf{n}}_\mathbf{i})$ is not affected by the underlying geometry and depends solely on the pattern. A small error is introduced in cases where the vertex density is not sufficient, but the angles θ_i remain a good descriptor of local features.

1. *Laplacian Smoothing:* The smoothing of the mesh is an iterative procedure which adjusts the position of each vertex based on the position of its neigborhood. The process is described by the following recursive equation:

$$\mathbf{p_i^n} = \mathbf{p_i^{n-1}} + \lambda \mathcal{L}(\mathbf{p_i^{n-1}})$$

$$\mathcal{L}(\mathbf{p_i^{n-1}}) = \frac{\sum\limits_{j \in \mathcal{N}_i} \mathbf{p_j^{n-1}} \cdot w_{ij}}{\sum\limits_{j \in \mathcal{N}_i} w_{ij}} - \mathbf{p_i^{n-1}}$$

$$w_{ij} = \sqrt{(\mathbf{p_j} - \mathbf{p_i}) \cdot (\mathbf{p_j} - \mathbf{p_i})^T}$$

   Authors set 30 iterations with a smoothing factor $\lambda = 0.7$ in order to erase the pattern from the meshes. An example of the final output of this step is shown in Figure 13(a).

2. *Theta Calculation:* The normal vectors of the original and the smooth models are computed as well as the angles between them. As can be seen from the visualization in Figure 13(c), the angles outline the local features with great precision. Thus, it can be concluded that the process can be generalized to more complex shapes than planes and cylinders.

3. *Surface Segmentation:* In order to distinguish areas containing pure textures from those with little to no texture, authors use the magnitude of the *saliency* s_i of the vertices, similarly to [25]. In particular, points with small saliency are considered to lie on flat areas. Points with high saliency are either part of a texture or lie in areas with significant geometric deformation. Generally, the latter points are few and far between, so they are not taken into consideration. More precisely, for each vertex $\mathbf{v_i}$ of the mesh, a patch of the 20 closest geometrical neighbours (including $\mathbf{v_i}$) is

created, together with a matrix $\mathbf{N_i} = (\mathbf{n_1^T}\ \mathbf{n_2^T}\ ...\ \mathbf{n_N^T})$ of their normals. Afterwards the co-variance matrix is formed as:

$$\mathbf{R}_i = \mathbf{N}_i^T \mathbf{N}_i \in \mathbb{R}^{3\times 3}\ \forall\ i = 1, \cdots, n \tag{7}$$

Decomposing the covariance matrix leads to $\mathbf{R}_i = \mathbf{U\Lambda U^{-1}}$

Finally, the saliency value of each vertex is computed as

$$s_i = \frac{1}{\sqrt{\lambda_{i1}^2 + \lambda_{i2}^2 + \lambda_{i3}^2}}\ \forall\ i = 1, \cdots, n \tag{8}$$

where $\lambda_{1i},\ \lambda_{2i},\ \lambda_{3i}$ are the elements of Λ. Finally, k-means is used to cluster the points in the two aforementioned categories: with or without texture. The points that belong to the cluster with the smallest centroid are considered to be part of flat areas (4 centroids were used in total). The points have now been labeled but they are randomly scattered along the surface of the mesh. A density based clustering helps unify them into large textureless areas. A variation of the DBSCAN algorithm [17] is used to find and connect neighboring flat points. In this variation the connectivity of the mesh is used to define a one-ring topological neighborhood instead of a geometrical one. An arbitrarily large threshold of points (in this case 1000) per area ensures that only large areas are classified as flat. The segmentation result is visualized in Figure 13(d).

4. *Feature Vector Extraction:* Finally, a feature vector needs to be computed for each model. The feature vector is a concatenation of 2 histograms H_1, H_2(see Table 1) multiplied by the weights w_1, w_2 defined as follows:

$$w_1 = \frac{\text{number of points in the "flat" areas}}{\text{total number of points}} \tag{9}$$

$$w_2 = \frac{\text{number of points in the "texture" areas}}{\text{total number of points}} \tag{10}$$

Probability normalization is applied to both histograms to bring the values between models to the same order of magnitude. It is important to note that while flat areas may also contain some minor characteristics of the texture (see Figure 13(d)), they have to be taken into consideration during feature vector extraction. Finally, the feature vector is equal to:

$$FV = [w_1 \cdot H_1,\ w_2 \cdot H_2]; \tag{11}$$

The distances between each model are computed using the Manhattan distance, which makes the calculation very efficient computation-wise. Finally, the distance matrix obtained by comparing all the models is computed. While the method itself will take a significant amount of time to finish, it is highly parallelizable.

Histogram	Description	Value range
H1	angles in the "flat" areas	$[0, \frac{\pi}{4}]$
H2	angles in the "texture" areas	$[0, \frac{\pi}{2}]$
H3	st. dev. of the angles in the "flat" areas	$[0, 0.5]$
H4	st. dev. of the angles in the "texture" areas	$[0, 0.5]$

Table 1: Description and parameterization of the Histograms used for the feature vector of the SRNA method. For every histogram a constant number of 30 uniformly sampled bins has been used.

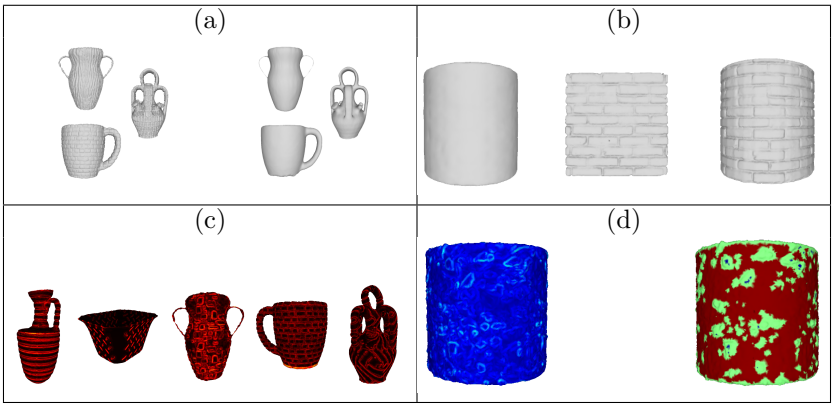


Figure 13: Overview of the steps of the SRNA method. (a): on the left, the original models with texture, while on the right the smoothed models without texture. (b): On the left, a smooth cylinder; on the center, a plane with texture; on the right, a cylinder with texture. (c): Theta angles visualization on different models. (d): Thetas (on the left) and segmentation (on the right) of the same model.

5. *Neighborhood angle standard deviation:* The method described so far only depends on the set of θ angles of a mesh. As a variation of the previous steps, the authors included some neighborhood information in the feature vector. If two textures display similar angles but differ in their spatial distributions they would otherwise be classified as the same. This extra bit of information can help distinguish between them and improve the accuracy of the method. However, computing a complex, rotation-invariant spatial descriptor is by no means an easy task, so authors overcome this problem by using the standard deviation of angles in small topological neighborhoods (1 rings). If the normals of that area have a common orientation the value will be small, whereas irregular areas will display much larger values. Two more histograms H_3, H_4 (see Table 1) with weights $w_3 = w_1, w_4 = w_2$ for the flat and texture areas are added to produce the final feature vector. A distance matrix is finally computed as described in the step 4.

4.6 Mesh Local Binary Pattern (meshLBP*), meshLBP-Sobel and meshLBP-Sharpen by Claudio Tortorici, Naoufel Werghi, Ahmad Shaker Obeid and Stefano Berretti

This method comprises four stages: (i) the extraction of patches from the object surface, (ii) the regularization of the patches tessellation, (iii) the computation of the descriptors, and finally, (iv) the generation of the dissimilarity matrix. Being based on a local patch analysis, this approach is intrinsically de-correlated from the global shape of the surface. These four steps are described down below.

1. *Patches extraction from the objects surface:* Up to six points are selected on the surface, obtained intersecting the mesh surface with the three eigenvectors of the covariance matrix centered at the center of mass of the object (see Figure 14(a)). This process, depending on the object shape, can detect 4 to 6 points on the mesh surface. Around each of these points a region is sampled (called patch), selecting only the vertices of the mesh within a given geodesical radius. Among those, the three patches with the largest ratio $\frac{e_2}{e_3}$, where e_2 and e_3 are respectively the second and third eigenvalues associated to the eigenvectors of the covariance matrix of the patch. An example of the final outcome of this step is shown in Figure 14(b).

2. *Regularization of the patches tessellation:* The three patches are then resampled by projection (PR) [49]. At first, (a) PCA is used to determine the two main axes of the sample that define a 2D plane of projection; (b) a uniform 2D grid of points is generated, which is then triangulated using the Delaunay algorithm; (c) the points on the grid are projected back to the mesh surface using interpolation, while keeping their triangulation intact.

3. *Descriptor computation:* Using the ORF structure (see Figure 15), it is possible to compute LBP patterns and perform convolution-like operations locally and directly on the mesh. In particular, the authors use the ORF (as in [45]) as a tool to operate convolution on the mesh manifold, and represent them as Convolution Binary Pattern [45]. Leveraging on the ordered structure of the ORF, the authors redefine the convolution operator between a mesh $\mathcal{M}$ and a filter $\mathcal{F}$ in polar coordinates as follows:

$$(\mathcal{M} * \mathcal{F}) = \sum_{r} \sum_{\theta} m_{r,\theta} \cdot f_{r,\theta} \ , \tag{12}$$

   where $m_{r,\theta}$, and $f_{r,\theta}$ are, respectively, a scalar function computed on the mesh and the filter values, both at radius r and angle θ. Subsequently, the response to the filter is used as input for the MeshLBP descriptor, thus obtaining a convolution binary pattern. Finally, the local descriptors are accumulated on a histogram computed over the entire patch surface.

4. *Dissimilarity matrix computation:* To compare two models, authors compare all their patches together in a pair-wise manner using Bhattacharya

distance. The dissimilarity between the two models is obtained by accumulating such distances.

Three runs have been submitted, changing the descriptor computed directly on the mesh manifold. The descriptors used in the runs are listed in Section 6.1

4.7 Correspondence matching based on kd-tree Fast Library for Approximate Nearest Neighbors (kd-tree FLANN) by Yoko Arteaga and Ramammorthy Luxman

This method is based on using the kd-tree Fast Library for Approximate Nearest Neighbors (FLANN [18]) correspondence matching to match the query of the other objects in the database. FLANN stands for Fast Library for Approximate Nearest Neighbors and it is a method used for evaluating the correspondence between two objects, by finding the distance between the extracted features. For this context, the kd-tree radius is set to be 0.15. This method works essentially in two steps: a pre-processing step that only extracts the surface information and the application of the kd-tree FLANN method on such information. For each patch, the authors detect points that are suitable for effective descrip-

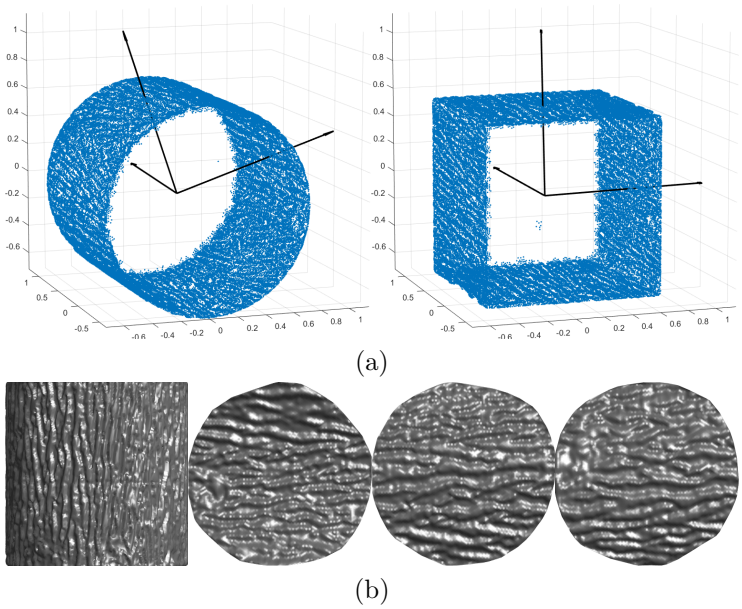


Figure 14: Patches extraction for the meshLBP-* method. (a) Intersection between the eigenvectors of the covariance matrix with the object surface to select the candidate points for the patch extraction. (b) A sample model is shown on the left, and the three extracted patches are reported on the right.

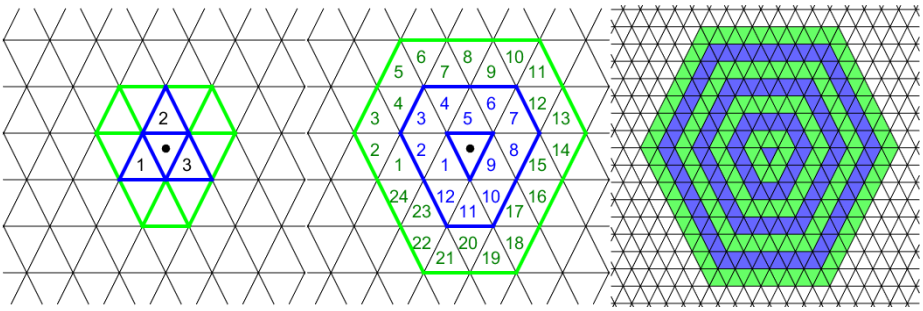


Figure 15: ORF ordered structure, showing its construction procedure, the ordered rings and its extension to multiple rings.

tion and matching, using uniform sampling keypoints detection method. Then, the local feature descriptor SHOT352 [44] is applied for each of the detected keypoints. The obtained features are then matched using KD Tree FLANN correspondence matching method. Notice that, instead of analysing the global shape, this method matches the features from the representative patches in order to de-correlate relief from shape. The criteria for choosing the patches is that the curvature must be minimum so it belongs to a flatter section of the object.

The pre-processing step is done to speed up the retrieval and to ensure only texture information is used. First, 400 points within the object are chosen at random. From each of the 400 points, their 400 nearest neighbors are found. Then, the mean curvature and the normals of each of the 400 patches are found. The final patch used as the representative of the object is selected as follows: it must be the one with the lower mean curvature and the greater mean variance of the normals. That is because, if the mean curvature is low, the patch belongs to a flat area of the object with the least curvature from its global shape, while the highest mean variance of normals implies that this area has the highest distribution of peaks and valleys in the sample, i.e. more texture. An example of the extracted representative is shown in Figure 16.

The kd-tree FLANN method is used to match the model representatives. Each entry of the dissimilarity matrix is equal to the inverse of the number of matches obtained as results from the kd-tree FLANN matching. If no matches are found, the value is set to 1.

5 Evaluation measures

We selected different evaluation measures for this SHREC track. The combination of these measures gives us a global view of the various methods, highlighting various properties (goodness of the method per model, class, overall based on multiple criteria). These measures are well known performance measures in information retrieval [37, 7] and many of them are already used in related SHREC

Figure 16: On the left, the final point cloud obtained after the conversion used for the kd-tree FLANN method. On the center, the representative patch (in blue) no the model. On the right, final extracted patch.

tracks [31, 11]. To better understand which measure does what, in the following we describe the evaluation measures we are going to use.

Nearest Neighbor (NN), First tier (FT), Second tier (ST). These measures checks the fraction of models in the query class also appearing within the top k retrievals [38]. In the case of NN, k is 1 and corresponds to the classification rate if the nearest neighbor classifier would be performed. Given a class of $|C|$ elements, k is $|C|-1$ for the FT and k is $2*(|C|-1)$ for the ST. Higher values of the NN, FT and ST measures indicate better matches. These measures range in the interval $[0,1]$.

Normalized Discounted Cumulated Gain (nDCG). This measure is based on the assumption that relevant items are more useful if appearing earlier in the list of the retrieved items. The nDCG is based on the graded relevance of a result with respect to the query. Then, the value is normalized with respect to the ideal outcome of that query.

Average precision-recall curves, mAP and e-Measure (e). Precision is the fraction of retrieved items that are relevant to the query. Recall is the fraction of the items relevant to the query that are successfully retrieved. By plotting the precision value with respect to the recall value we obtain the so called recall vs. precision curve: the larger the area below such a curve, the better. In particular, the precision-recall curve of an ideal retrieval system would result in a constant curve equal to 1. For each query, we have a precision-recall (PR) curve. In our context, results are evaluated on the mean of all the PR curves. The *mean Average Precision (mAP)* corresponds to the area between the horizontal axis and the average precision-recall curve and ranges from 0 to 1. The higher, the better. The e-Measure (e) derives from the precision and recall for a fixed number of retrieved results (32 in our settings), [37]. For every query, the e-Measure considers the first 32 retrieved items and is defined as $e = \frac{1}{\frac{1}{P}+\frac{1}{R}}$, where P and R represent the precision and recall values over those results, respectively.

Confusion matrix. To each run we associate also a confusion matrix CM, that is, a square matrix whose order is equal to the number of classes in the dataset. For a row i in CM, the element $CM(i,i)$ gives the number of items which have been

correctly classified as elements of the class i. The elements $CM(i,j)$, with $j \neq i$, count the items of the class i which have been misclassified and j corresponds to the class in which they were wrongly classified. An ideal classification system should be a diagonal matrix. The sum $\sum_j CM(i,j)$ equals the number of items in the class i. Generally, the confusion matrix is non-symmetric.

Tier images. Similar to the confusion matrix, the tier image visualizes the matches of the NN, FT and ST. The value of the element $T(i,j)$ is: *black* if j is the NN of i, *red* if j is among the $(|C|-1)$ top matches (FT) and *blue* if j is among the $2(|C|-1)$ top matches (ST), where $|C|$ is the number of elements of the class C. The models of a class are grouped along each axis so it is easier to interpret. With this configuration, the optimal tier image clusters the black/red square pixels on the diagonal.

Receiver Operating Characteristic (ROC) curve and AUC value. ROC curves are largely used to evaluate the classification performance of a method and are suitable to assess retrieval issues, too. The ROC curve shows the ratio between False Positive Rate and True Positive Rate for each model at different classification thresholds. In our scenario, the classification thresholds are the number of models in each class (20) multiplied by a scalar value that goes from 1 to the number of classes in the dataset (11). The higher the curve is the better. A quick comparison between the methods based on the ROC curves can be derived also by the AUC value (namely the *area under curve* value), which is the measure of the area under the ROC curve. The higher this value is, the better. Anyway, note that an AUC value of 0.5 means that the corresponding method is not able to classify the models at all. In this work, we consider the mean of all ROC curves.

6 Description and evaluation of the submitted runs

In this Section, the settings of the runs submitted for evaluation are detailed. Their outcome is presented with respect to the performance measures described in Section 5.

6.1 Run settings

In the following, the parameter settings of the runs submitted are listed. If an author sent a single run, the setting are those described in Section 4. The same happens if the runs of a single method differ more than "just" different parameters.

- *APPFD-FK*: two runs (APPFD-FK(run1) and APPFD-FK(run2) respectively) use a mono-dimensional, $[0,1]$ *normalized* histogram with 35 bins for each one of the feature-dimensions in the APPF; these histograms are concatenated to yield a final 6 x 35 = 210-dimensional local descriptor

for each LSP or key-point. In APPFD-FK(run3), the six-dimensional features in the APPF are discretized into a multi-dimensional histogram with 5 bins for each feature-dimension, the histogram is then flattened and normalized to give a $5^6 = 15625$-dimensional local descriptor for each LSP or key-point.

- *OH*: The authors sent three runs for this method, changing the number of bins of the histogram or the metric used for the dissimilarity matrix computation (or both). In particular,
  - OH(run1): Number of bins: N = 200 - metric: modified L1 norm
  - OH(run2): Number of bins: N = 200 - metric: modified cosine-similarity.
  - OH(run3): Number of bins: N = 128 - metric: modified L1 norm.
- *DFE*: The runs differ in the models used for transfer learning. The models used in each run are listed in the following.
  - DFE(run1):
    - model 1: DenseNet201(layer pool3_pool) + Global Average Pooling
    - model 2: DenseNet201(layer pool4_pool) + Global Average Pooling
    - model 3: DenseNet169(layer pool3_pool) + Global Average Pooling

    The authors ensemble the three models above with the ratio of weights: $model1 : model2 : model3 = 2 : 1 : 1$, using the Cosine distance as metric.
  - DFE(run2): Same model settings as DFE(run1), using Euclidean distance as metric.
  - DFE(run3):
    - model 1: ResNet152(layer conv4_block36_out) + Global Average Pooling
    - model 2: ResNet152(layer conv5_block3_out) + Global Average Pooling
    - model 3: ResNet101(layer conv4_block23_out) + Global Average Pooling

    The authors ensemble the three models above with the ratio of weights: $model1 : model2 : model3 = 2 : 1 : 2$, using the Cosine distance as metric.
- *DPML*: The runs differ in the way the patches generated are pre-processed and in the use of data augmentation.

  - DPML(run1): once obtained the patches authors uniformed them cropping to obtain images of size 231×231. No data augmentation has been used in this run.
  - DPML(run2): Same model as in Run 1. This time instead of cropping patches, authors padded them with zero-values to the size of the biggest patch that is 836×836. Furthermore, during training data augmentation was applied rotating input image with random angles and also flipping vertically and horizontally.

- *PointNet+SQFD*: All the runs used the following clustering parameters: $\lambda = 0.3, \beta = 0.4$ and $Nm = 10$. The other settings are listed in the following:
  - PointNet+SQFD(run1): number of patches $p = 100$ of diameter $diam = 0.1$ of the diagonal of the bounding box of the shape.
  - PointNet+SQFD(run2): number of patches $p = 200$ of diameter $diam = 0.05$ of the diagonal of the bounding box of the shape.
  - PointNet+SQFD(run3): number of patches $p = 500$ of diameter $diam = 0.025$ of the diagonal of the bounding box of the shape.

  In all the submitted runs, $\alpha = 0.9$ and d is the L_2 distance.

- *SRNA*: Two runs have been proposed for this method:
  - SRNA(run1); this run corresponds to the outcome of the first four steps of the method described in Section 4.5;
  - SRNA(run2): this run corresponds to the variation of the SRNA method that includes also neighbour information as described in the step five of Section 4.5.

- *meshLBP-**: the descriptors used in each of the three runs submitted are:
  - (meshLBP-so): Sobel filter;
  - (meshLBP-sh): sharpen filter;
  - (meshLBP): MeshLBP.

6.2 Results

Table 2 summarizes the performances of all the twenty runs submitted for evaluation, namely each column of the Table reports the label of each run, the Nearest Neighbour (NN), the First Tier (FT), the Second Tier (ST), the normalized Discounted Cumulated Gain (nDCG), the e-measure (e) and the AUC value, respectively. The best performances for each measure are highlighted in bold. Many methods achieve good or very good performances. For example, 4 methods have an NN value above the level of 0.9, i.e. they have a classification rate above 90%. Similarly, the same 4 methods have the mAP value greater

	NN	FT	ST	mAP	nDCG	e	AUC
APPFD-FK(run1)	0.186	0.204	0.332	0.235	0.523	0.211	0.672
APPFD-FK(run2)	0.132	0.186	0.299	0.212	0.497	0.192	0.632
APPFD-FK(run3)	0.186	0.192	0.318	0.228	0.507	0.203	0.682
OH(run1)	0.791	0.406	0.567	0.470	0.737	0.377	0.817
OH(run2)	0.750	0.374	0.517	0.418	0.709	0.341	0.779
OH(run3)	0.714	0.405	0.575	0.469	0.732	0.382	0.818
DFE(run1)	0.982	0.920	1.000	0.930	0.974	0.715	0.987
DFE(run2)	0.982	0.913	1.000	0.926	0.973	0.714	0.986
DFE(run3)	0.982	0.865	1.000	0.896	0.963	0.693	0.980
DPML(run1)	0.900	0.836	0.990	0.868	0.941	0.686	0.974
DPML(run2)	0.982	0.887	0.992	0.912	0.968	0.690	0.978
PointNet+SQFD(run1)	0.095	0.095	0.184	0.168	0.440	0.113	0.569
PointNet+SQFD(run2)	0.077	0.099	0.203	0.171	0.442	0.122	0.582
PointNet+SQFD(run3)	0.173	0.119	0.225	0.190	0.470	0.137	0.605
SRNA(run1)	0.905	0.493	0.670	0.548	0.802	0.447	0.869
SRNA(run2)	0.923	0.494	0.683	0.563	0.811	0.453	0.882
meshLBP-so	0.909	0.631	0.764	0.687	0.872	0.516	0.870
meshLBP-sh	0.895	0.601	0.759	0.656	0.853	0.522	0.875
meshLBP	0.905	0.671	0.832	0.726	0.884	0.570	0.909
kd-tree FLANN	0.686	0.312	0.424	0.359	0.656	0.283	0.690

Table 2: Nearest Neighborhood, First Tier, Second Tier, mAP, nDGC, e-measure and AUC value of all the submitter runs. Values goes from 0 (red), to 1 (green). The higher the value is, the better the method performs.

than 0.7 and the nDCG greater than 0.8. Also, note that 2 methods have the ST score above 0.99 which, having all the classes 20 models each, means that the models with the same 3D texture as a query are generally found within the first 39 retrieved models, with very few exceptions.

For a better visual comparison of the methods, only the Confusion Matrix and Tier Image of the best run of each method are reported in Figure 18 and Figure 19 respectively. For completeness, the Confusion matrices and the Tier images of the all the runs submitted are listed in the Appendix. Precision-Recall plots of the best run for each method are shown in Figure 17. Similarly, only the ROC curves of the best runs are shown in Figure 20. As also reflected by the area under the ROC curve, methods with AUC greater than 0.97 provide a better classification than other methods. For completeness, the PR plots and the ROC curves of the all the runs submitted are listed in the Appendix. This more complete overview of the runs highlights that the performances of a method show the same trend for the different runs, with small qualitative variations between the different parameter choices.

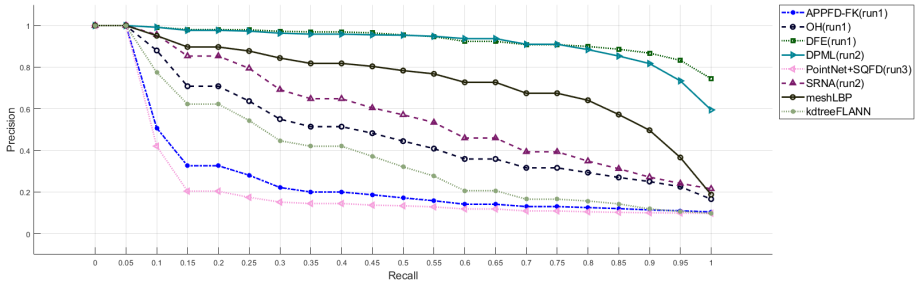


Figure 17: Overview of Precision-Recall plots of the best run for each method.

7 Discussions and concluding remarks

Overall, the best performances are obtained by the DFE method, which uses a pre-trained neural network. We observe that the NN, FT and ST scores for the methods based on transfer learning do not change significantly. This fact suggests that, if they have success, these methods have a larger capability of ranking the models that contains a texture similar to the query at the beginning of the list of the items retrieved, while the other methods drop around 0.3 from NN to FT. However, also methods that do not use learning techniques perform well (like the meshLBP, OH and SRNA). We notice that these methods are all based on feature vectors. Some methods share some background, for instance, the meshLBP-so run and the OH methods use of the Sobel Filter. However, among the three meshLBP-based runs submitted to this track, the best performances are reached by the meshLBP run that is based on convolution-like operations extended to a triangle mesh.

A common characteristic of most of methods is the sampling of one or more representative patches as a pre-processing step. It consists of a single patch (like in the case of the DFE, OH, DPML, kd-tree FLANN runs) or multiple ones (like in the APPFD-FK, PointNet+SQDF, SRNA, meshLBP runs). In general, the selection of a single patch seems to lead to good results with the exception of the SRNA and meshLBP methods, which compute a more statistical approach on the representative patches.

Methods that convert the model into point clouds (APPFD-FK) or that are based on CNNs trained on point clouds (PointNet) seem to be sub-optimal for this task. Probably these methods lose information on local details (for instance, the sampling process in the APPFD-FK focuses on the representation of the global geometry) and do not capture the subtle geometry and structure variations of local patterns and reliefs. On a similar note, the authors of the kd-tree FLANN method suggest that the performance of their methods will probably be improved by considering a smaller representative patch. With the current size of the patch, the global geometry of the model is still kept in consideration and it biases the results. This fact highlights the importance of analysing a surface with reliefs by local approaches (but that are robust to

noise).

From the Confusion Matrices we observe there is not a class (i.e. a realistic geometric relief that corresponds to a real texture) that is more complex to deal with at all. On the other hand, Tier Images highlight that some methods (DFE and meshLBP in particular) tend to confuse class 10 (straight horizontal lines with some thing double lines) and class 2 (just straight lines) or class 4 (bricks). Indeed, all these classes have a set of horizontal and parallel lines which lead to some uncertainty in the classification (especially classes 10 and 2).

In conclusion, we have presented the results of the SHREC'20 contest track on "Retrieval of surface patches with similar geometric reliefs". The number of runs (twenty) and methods (eight) is significantly numerous and show the increasing efforts of the community in the effective characterization of all the aspects of a surface. The runs and the methods submitted to this track present a satisfactory variety, in terms of the diversification of the approach followed (feature-based and learning-based methods) and the type of description chosen (global vs local descriptions). Several methods use a transfer learning approach based on pre-trained, image-based neural networks. For instance, the best performances are obtained by the DFE method, which follows such a strategy.

With respect to the methods submitted to similar, previous SHREC contests, we can observe the rise of the machine learning based approaches specifically designed for and/or adapted to this track task. A future direction of investigation is to deepen the analysis of the performances of methods based on learning. To this end, it will be necessary to create larger data collections, opportunely equipped with a training set of models, even if the application of reliefs to a surface is not trivial. Indeed, at the moment it requires some manual cleaning of the models, in particular in correspondence of high curvature features like handles.

The way models are analysed by most methods, that is a local conversion of the surface into a kind of texture image, helps in removing the influence of the underlying surface from the reliefs. Still, corresponding to models that can be manufactured, the surfaces of the models proposed in this benchmark can be locally projected in a plane and therefore in an image. Further research is needed to deal with more challenging models and how these methods work on models with a more complex geometry and/or how they could be patched to deal with them.

Overall, this contest has received a good number of satisfactory solutions that highlight the progress of recent years in the field of geometric pattern retrieval. As a future research direction we envisage an increase of interest in the more complex task of pattern recognition on surfaces, i.e., addressing a problem similar to the challenge proposed in [12], where the models were only partially covered by none, one, or many patterns. The goal of that track was to identify, from a set of sample patterns, if and where the same pattern was located on each model. At the time of that track [12] there were no satisfactory solutions. In the near future, in the light of the progress achieved in the pattern retrieval problem and the progress made in the field of transfer learning, it would become interesting to understand what can be exploited also in the field

of pattern recognition, too.

Acknowledgements

The authors thank the 3DOR 2020 Workshop and Program Chairs for helping us in the organization of our contest despite the current COVID-19 pandemic. We also thank the anonymous reviewers for providing constructive comments on earlier drafts of the manuscript, which helped us to improve and clarify this work. This study was partially supported by the CNR-IMATI projects DIT.AD004.100 and DIT.AD021.080.001. Research for the team from University of Science, Ho Chi Minh city, Vietnam is supported by Vingroup Innovation Foundation (VINIF) in project code VINIF.2019.DA19. The team Y. Arteaga and R. Luxman research is funded by the Horizon 2020 programme of the European Union Grant #813789. The work of Ivan Sipiran has been supported by Proyecto de Mejoramiento y Ampliación de los Servicios del Sistema Nacional de Ciencia Tecnología e Innovación Tecnológica(Banco Mundial, Concytec), Nro. grant 062-2018-FONDECYT-BM-IADT-AV. The work of Benjamin Bustos is funded by the Millennium Institute Foundational Research on Data (IMFD).

References

[1] MIT CSAIL Textured Models Database, 2008.

[2] Github - Fisher Vector, Python. `https://gist.github.com/danoneata/9927923/`, 2014. Accessed: 2020-03-12.

[3] Publish & find 3D models online. https://sketchfab.com/, 2020.

[4] Texture Haven. `https://texturehaven.com/`, 2020. Accessed: 2020-04-23.

[5] Turbosquid. https://www.turbosquid.com/3d-model, 2020.

[6] D. H. K. Azad and A. Deepak. Query expansion techniques for information retrieval: a survey. *Inf. Process. Manag.*, 56:1698–1735, 2017.

[7] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[8] C. Beecks, M. S. Uysal, and T. Seidl. Signature quadratic form distances for content-based similarity. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, page 697–700, New York, NY, USA, 2009.

[9] S. Biasotti, A. Cerri, M. Abdelrahman, M. Aono, A. B. Hamza, M. El-Melegy, A. Farag, V. Garro, A. Giachetti, D. Giorgi, and et al. Retrieval and classification on textured 3d models. In 7^{th} *EG Workshop on 3D Object Retrieval*, page 111–120. Eurographics Association, 2014.

[10] S. Biasotti, A. Cerri, M. Aono, A. B. Hamza, V. Garro, A. Giachetti, D. Giorgi, A. Godil, C. Li, C. Sanada, et al. Retrieval and classification methods for textured 3D models: A comparative study. *The Visual Computer*, 32(2):217–241, 2016.

[11] S. Biasotti, E. Moscoso Thompson, M. Aono, A. B. Hamza, B. Bustos, S. Dong, B. Du, A. Fehri, H. Li, F. A. Limberger, and et al. Retrieval of surfaces with similar relief patterns: Shrec'17 track. In 10^{th} *EG Workshop on 3D Object Retrieval*, 3Dor '17, page 95–103. Eurographics Association, 2017.

[12] S. Biasotti, E. Moscoso Thompson, L. Barthe, S. Berretti, A. Giachetti, T. Lejemble, N. Mellado, K. Moustakas, I. Manolas, D. Dimou, C. Tortorici, S. Velasco-Forero, N. Werghi, M. Polig, G. Sorrentino, and S. Hermon. Recognition of Geometric Patterns Over 3D Models. In 11^{th} *EG Workshop on 3D Object Retrieval*, pages 71 – 77. The Eurographics Association, 2018.

[13] A. Cerri, S. Biasotti, M. Abdelrahman, J. Angulo, K. Berger, L. Chevallier, M. El-Melegy, A. Farag, F. Lefebvre, A. Giachetti, H. Guermoud, Y.-J. Liu, S. Velasco-Forero, J. Vigouroux, C.-X. Xu, and J.-B. Zhang. SHREC'13 Track: Retrieval on Textured 3D Models. In 6^{th} *EG Workshop on 3D Object Retrieval*, pages 73–80. The Eurographics Association, 2013.

[14] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(36):1109–1135, 2010.

[15] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, page 539–546, USA, 2005. IEEE Computer Society.

[16] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 3606–3613, Washington, DC, USA, 2014. IEEE Computer Society.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[18] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, Sept. 1977.

[19] A. Giachetti. Effective characterization of relief patterns. *Computer Graphics Forum*, 37(5):83–92, 2018.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[21] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger. Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[22] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[23] N. Kanopoulos, N. Vasanthavada, and R. L. Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988.

[24] W. Mathworld. Dot Product. `http://mathworld.wolfram.com/DotProduct.html`, accessed: 2019-10-15.

[25] E. Moscoso Thompson, G. Arvanitis, K. Moustakas, N. Hoang-Xuan, E. R. Nguyen, M. Tran, T. Lejemble, L. Barthe, N. Mellado, C. Romanengo, S. Biasotti, and B. FALCIDIENO. Feature Curve Extraction on Triangle Meshes. In 12^{th} *EG Workshop on 3D Object Retrieval*, pages 85–92. The Eurographics Association, 2019.

[26] E. Moscoso Thompson and S. Biasotti. Description and retrieval of geometric patterns on surface meshes using an edge-based lbp approach. *Pattern Recognition*, 82:1 – 15, 2018.

[27] E. Moscoso Thompson and S. Biasotti. Edge-based LBP Description of Surfaces with Colorimetric Patterns. In 11^{th} *EG Workshop on 3D Object Retrieval*, pages 1 – 8. The Eurographics Association, 2018.

[28] E. Moscoso Thompson and S. Biasotti. Retrieving color patterns on surface meshes using edgelbp descriptors. *Computers & Graphics*, 79:46 – 57, 2019.

[29] E. Moscoso Thompson, S. Biasotti, J. Digne, and R. Chaine. mpLBP: An Extension of the Local Binary Pattern to Surfaces based on an Efficient Coding of the Point Neighbours. In 12^{th} *EG Workshop on 3D Object Retrieval*, pages 9–16. The Eurographics Association, 2019.

[30] E. Moscoso Thompson, S. Biasotti, J. Digne, and R. Chaine. mplbp: A point-based representation for surface pattern description. *Computers & Graphics*, 86:81 – 92, 2020.

[31] E. Moscoso Thompson, C. Tortorici, N. Werghi, S. Berretti, S. Velasco-Forero, and S. Biasotti. Retrieval of gray patterns depicted on 3d models. In 11^{th} *EG Workshop on 3D Object Retrieval*, 3DOR '18, page 63–69. The Eurographics Association, 2018.

[32] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[33] A. Othmani, F. Torkhani, and J. Favreau. 3d geometric salient patterns analysis on 3d meshes. *CoRR*, abs/1906.07645, 2019.

[34] A. Othmani, L. F. L. Y. Voon, C. Stolz, and A. Piboule. Single tree species classification from terrestrial laser scanning data for forest inventory. *Pattern Recognition Letters*, 34(16):2144–2150, 2013.

[35] E. Otu, R. Zwiggelaar, D. Hunter, and Y. Liu. Nonrigid 3d shape retrieval with happs: A novel hybrid augmented point pair signature. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 662–668, 2019.

[36] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[37] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[38] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Shape Modeling International*, page 167–178, jun 2004.

[39] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. In *BMVC*, volume 2, page 4, 2013.

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[42] I. Sipiran, J. Loko, B. Bustos, and T. Skopal. Scalable 3d shape retrieval using local features and the signature quadratic form distance. *The Visual Computer*, 33:1571–1585, 2017.

[43] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[44] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ECCV'10, page 356–369, Berlin, Heidelberg, 2010. Springer-Verlag.

[45] C. Tortorici, N. Werghi, and S. Berretti. Extending LBP and Convolution-Like Operations on the Mesh. In *IEEE Int. Conf. on Image Processing (ICIP)*, pages 4479–4483. IEEE, sep 2019.

[46] E. Wahl, U. Hillenbrand, and G. Hirzinger. Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 474–481, 2003.

[47] N. Werghi, S. Berretti, and A. D. Bimbo. The mesh-LBP: A framework for extracting local binary patterns from discrete manifolds. *IEEE Trans. Image Processing*, 24(1):220–235, 2015.

[48] N. Werghi, C. Tortorici, S. Berretti, and A. D. Bimbo. Local binary patterns on triangular meshes: Concept and applications. *Computer Vision and Image Understanding*, 139:161–177, 2015.

[49] N. Werghi, C. Tortorici, S. Berretti, and A. Del Bimbo. Representing 3D texture on mesh manifolds for retrieval and recognition applications. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 07-12-June, pages 2521–2530. IEEE, jun 2015.

[50] Wikipedia. Dot Product. `https://en.wikipedia.org/wiki/Dot_product`, accessed: 2019-10-15.

[51] C. Yuksel. Sample elimination for generating poisson disk sample sets. *Computer Graphics Forum*, 34, 05 2015.

[52] M. Zeppelzauer, G. Poier, M. Seidl, C. Reinbacher, S. Schulter, C. Breiteneder, and H. Bischof. Interactive 3D segmentation of rock-art by enhanced depth maps and gradient preserving regularization. *J. Comput. Cult. Herit.*, 9(4):19:1–19:30, Sept. 2016.

[53] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

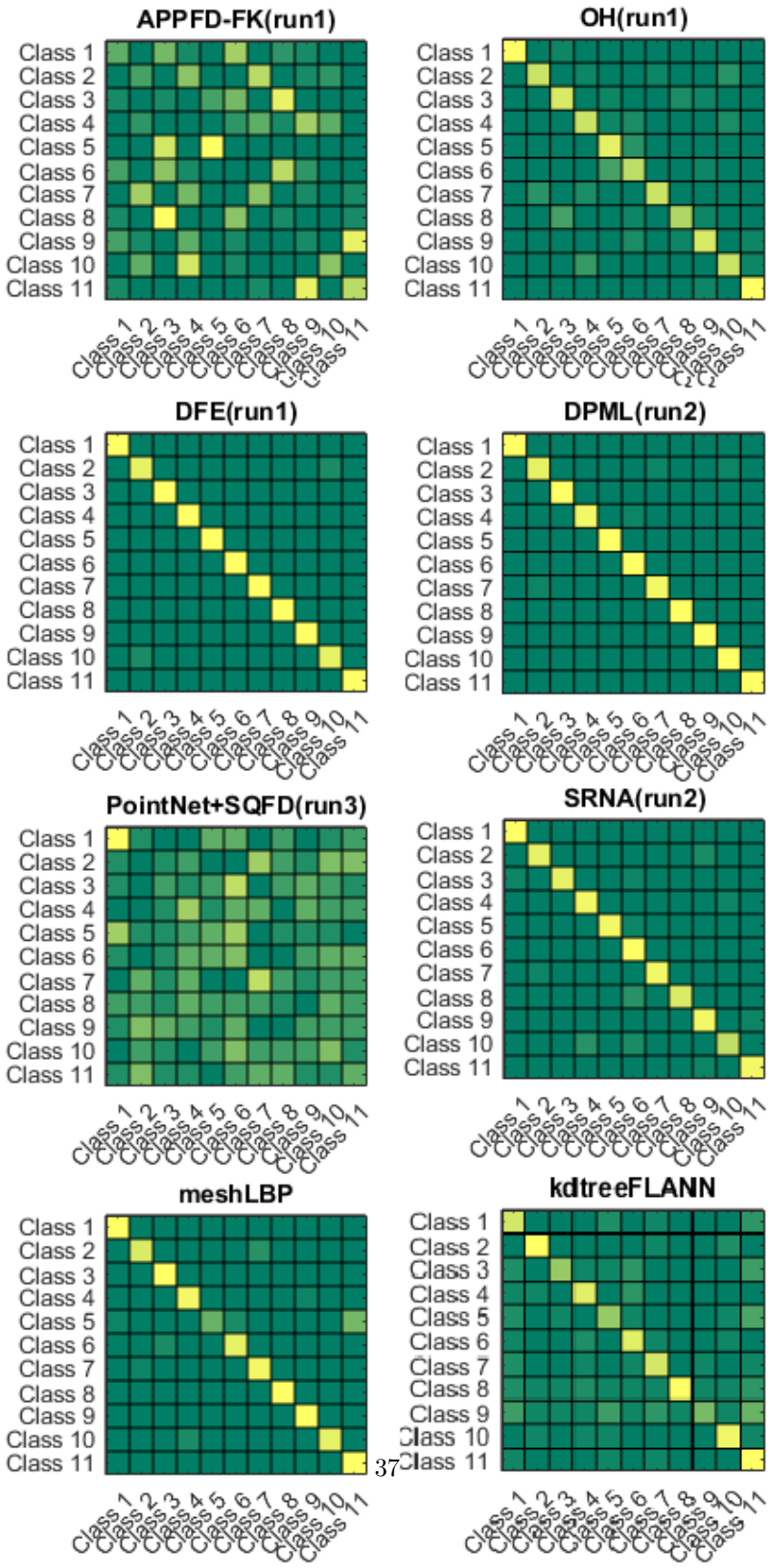


Figure 18: Overview of the confusion matrices of the best run for all the methods.

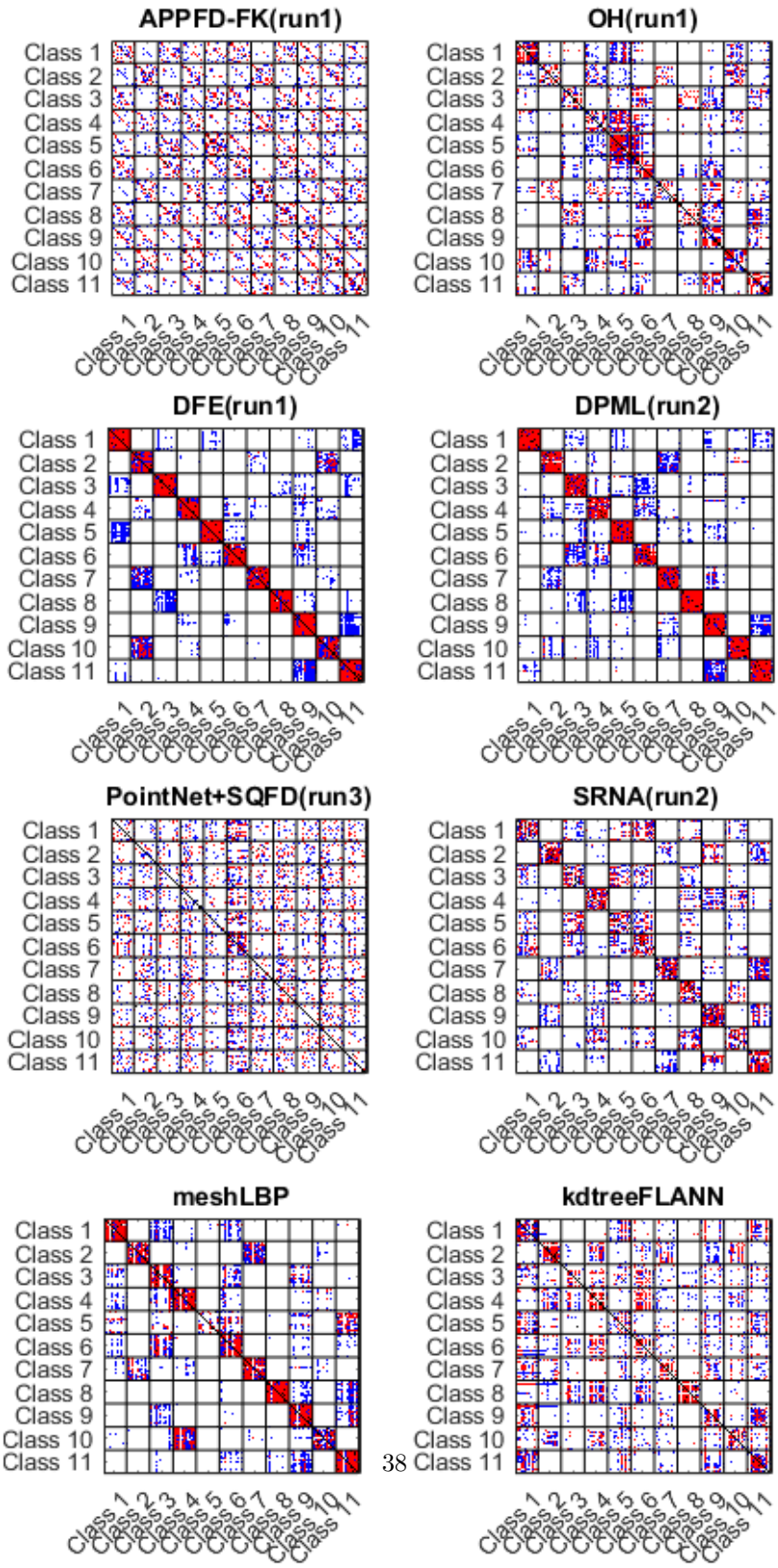


Figure 19: Overview of the tier images of the best run for all the methods.

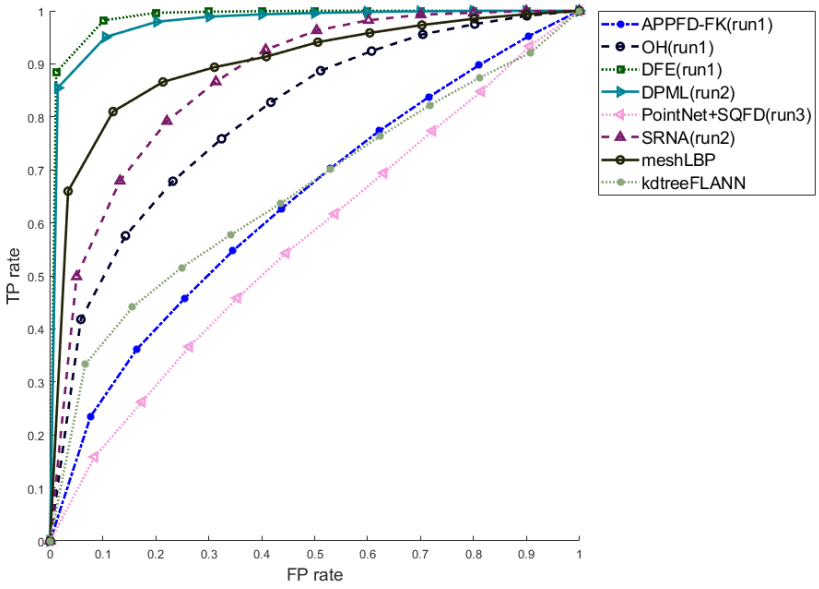
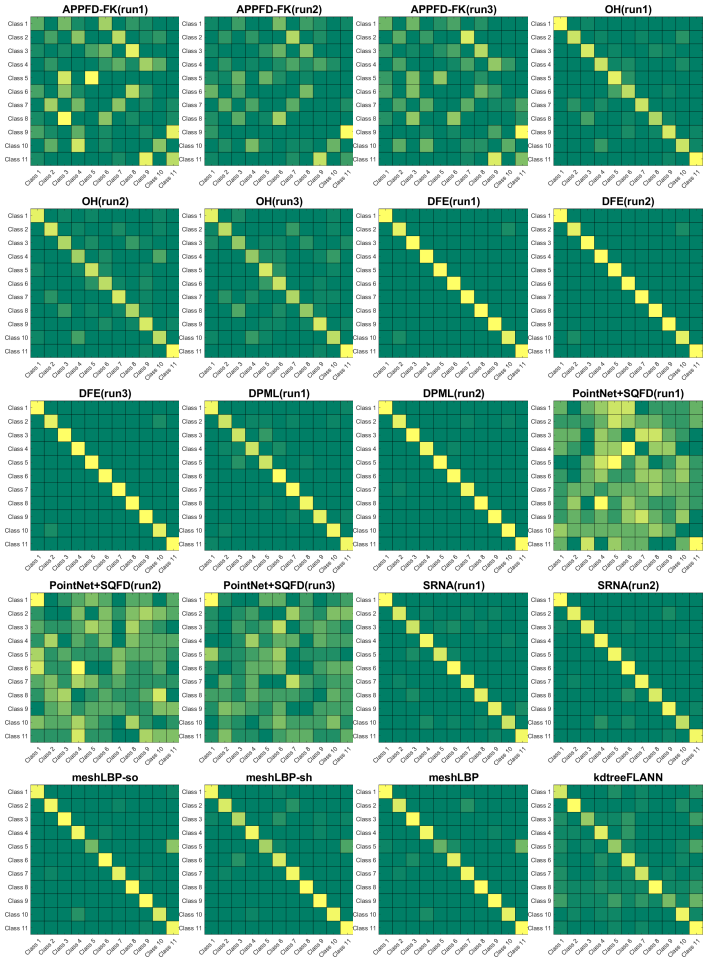
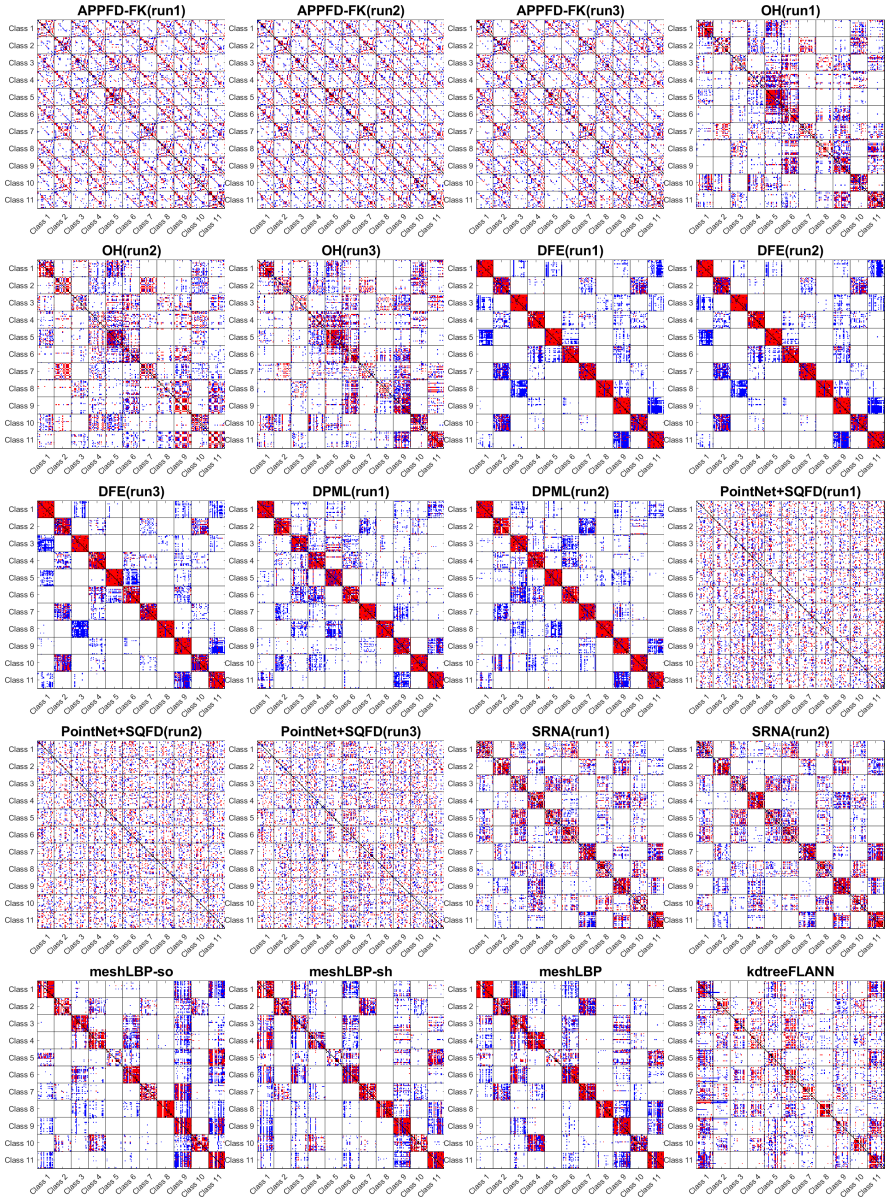


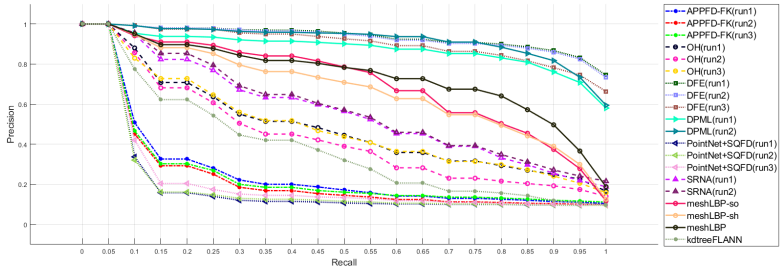
Figure 20: Overview of ROC curves of the best run for each method.

A Confusion Matrices, Tier Images, PR plots and ROC curves




APPFD-FK(run1)
APPFD-FK(run2)
APPFD-FK(run3)
OH(run1)
OH(run2)
OH(run3)
DFE(run1)
DFE(run2)
DFE(run3)
DPML(run1)
DPML(run2)
PointNet+SQFD(run1)
PointNet+SQFD(run2)
PointNet+SQFD(run3)
SRNA(run1)
SRNA(run2)
meshLBP-so
meshLBP-sh
meshLBP
kdtreeFLANN
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9
Class 10
Class 11

Precision-Recall plots



ROC curves